On a Dynamic Logic for Graph Rewriting

Mathias Winckel, Ralph Matthes

On a Dynamic Logic for Graph Rewriting *

Mathias Winckel and Ralph Matthes

Institut de Recherche en Informatique de Toulouse (IRIT)

{winckel, matthes}@irit.fr

Abstract. Initially introduced by P. Balbiani, R. Echahed and A. Herzig, this dynamic logic is useful to talk about properties on termgraphs and to characterize transformations on these graphs. Also are presented the deterministic labelled graphs for which the logical framework is designed.
This logic has been the starting point of a formal development, using the Coq proof assistant, to design a logical and algorithmic framework useful for verifyin and proving graph rewriting. The formalization allowed us to figure out some ambiguities in the involved concepts. This formalization is not the topic here but the clear view brought to us by the formal work, so the results will be expressed using the original mathematical objects of this logic.
Some problems of this logic are demonstrated, relatively to the representation of graph rewriting. Some are minor issues but some are far more important for the adequation between the formulas about graph rewriting and the actual rewriting systems. Invalidating some resulting propositions, solutions are given to reestablish the logical characterization of graph rewriting, which was the initial purpose.

Keywords. Dynamic Logic, Graph Rewriting, Adequation Issues

Key terms. Formal Method, Model

1 Introduction

Nearly every field of computer science uses graphs to represent data or the behavior of systems. Then, to get a higher level of dynamics, it uses rewriting in a more or less formal way to handle manipulation of such objects. In some fields, as in Formal Methods and Model-Driven Engineering, graphs are one of the main tools, having advanced methods to express and reason on these graphs is of great pertinance.

Modal logic allows to express relational properties naturally and, with Kripke semantics, is closely linked to graphs which are its models. Yet, instead of discussing graphs and rewriting using hyper-graphs as models, the transformed

* This work has been funded by the CLIMT project (ANR-11-BS02-016 of the French Agence Nationale de la Recherche)

graphs could be directly these models. Dynamic logic as defined by D. Harel offers by its modalities the possibility to express relations on models when they have some desired properties. In this spirit, introduced during ICGT 2010, "A Dynamic Logic for Termgraph Rewriting" [1] was proposed by P. Balbiani, R. Echahed and A. Herzig as a suitable dynamic logic to describe graphs and transformations on these graphs.

The termgraph lifts the concept of term to the more general one of graph. It was initially introduced to represent terms while having a simpler way to talk about recursion or sharing of sub-terms, what a tree-like structure doesn't allow to do easily.

Computer science involves data structures that are usually syntactically represented as simple terms. It is interesting to develop such a tool for more than just a graphic representation: using graph rewriting, research could be done on term rewriting with this rich and powerful layer of language of graph structure.

In the following, in *Section 2* we will present the type of graphs we use, then in *Section 3* the dynamic logic for graph rewriting, its syntax and its semantics. In *Section 4* the rewriting system is presented, and propositions to logically talk about it, but with issues to express these concepts with the logic as originally introduced. Graph homomorphisms, rewriting steps and application to a graph, with the definition of rewriting rules, matching of rules and normal forms, actually leads to some divergences between actual rewriting and its translation using the logic. *Section 4* also discuss and proposes some solutions for these original issues, for such an utilisation.

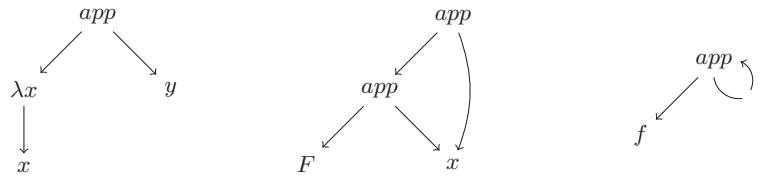
2 Termgraph And Rooted Termgraph

To illustrate terms representation as graphs from some typical λ-calculus, as shown in Figure 1, a classical tree-like term in (a), an example of argument sharing in (b), and eventually a representation for a fixpoint operator in (c), usually denoted Y for a function f [2]. One can easily unfold the recursion in the latter graph and get $Y\ f = f\ (Y\ f)$, which is what is expected from a fixpoint operator.

The graphs are deterministic and has labelled nodes and edges [3], but for convenience edge labels will be named features. A linear graph grammar is defined over a set of nodes $\mathcal{N}$, a set of labels Ω and a set of features $\mathcal{F}$ [1], by the following rules:

$$Node ::= n : \omega\ (f_1 \Rightarrow Node, \ldots, f_k \Rightarrow Node) \mid n : \bullet \mid n$$
$$\text{avec } n \in \mathcal{N}, \omega \in \Omega \text{ et } f_1, \ldots, f_k \in \mathcal{F}$$
$$TermGraph ::= Node \mid Node + TermGraph$$

Allowing one to define a node, with its label and its direct sons, a node without label or just a reference to an actual node with the first rules, and to define multiparts termgraphs with the others.



(a) *term*: $(\lambda x.x)\ y$ (b) *sharing*: $F\ x\ x$ (c) *recursor*: $Y\ f$

Fig. 1. Examples of term representation

In a less syntactic and maybe more semantic manner, a termgraph is defined as well by a structure.

$$(\mathcal{N}, \mathcal{E}, \mathcal{L}^{\mathcal{N}}, \mathcal{L}^{\mathcal{E}}, \mathcal{S}, \mathcal{T}) \text{ with}$$

- $\mathcal{N}$ a finite set of nodes
- $\mathcal{E}$ a finite set of edges
- $\mathcal{L}^{\mathcal{N}}$ a partial function from $\mathcal{N}$ to Ω, associating a node to its label
- $\mathcal{L}^{\mathcal{E}}$ a total function from $\mathcal{E}$ to $\mathcal{F}$, associating a edge to its feature
- $\mathcal{S}$ a source function from $\mathcal{E}$ to $\mathcal{N}$
- $\mathcal{T}$ a target function from $\mathcal{E}$ to $\mathcal{N}$

It's assumed that the previous definitions respect a determinism condition defined as

$$\forall\, e_1,\, e_2 \in \mathcal{E},\, \mathcal{S}(e_1) = \mathcal{S}(e_2) \,\wedge\, \mathcal{L}^{\mathcal{E}}(e_1) = \mathcal{L}^{\mathcal{E}}(e_2) \rightarrow e_1 = e_2$$

Rooted TermGraphs. In the following, for the need of the logic, the graphs will be an extension of the termgrah with a specific node pointed as its root.

$$(\mathcal{N},\ \mathcal{E},\ \mathcal{L}^{\mathcal{N}},\ \mathcal{L}^{\mathcal{E}},\ \mathcal{S},\ \mathcal{T},\ r) \text{ with the root } r \in \mathcal{N}$$

3 Dynamic Logic for Graph Rewriting

A modal logic is a propositional logic, extended with one or several modality operators. Dynamic logic [4] is a multi-modal logic, its modalities being actions and the possibility to express a choice, an iteration or the sequence of actions. Actions can be defined to express graph transformations by miscellaneous modalities.

3.1 Syntax of the Dynamic Logic

For given countable sets $\mathcal{F}$ and Ω, of features and labels (their respective elements being usually denoted $a, b, ...$ and $\omega, \pi, ...$), the rules defining the formulas and the actions of the syntax are the following. [1]

For an action α:

$\alpha ::= a$ for the navigation by $a \in \mathcal{F}$ from the root
$\mid U$ for changing the root to any node in the graph
$\mid n \mid \boldsymbol{n}$ for the creation of a node n, eventually setting it as the root
$\mid \phi?$ for the verification of the validity of a formula ϕ for the root
$\mid (\omega :=_l \phi) \mid (\omega :=_g \phi)$ for labeling a node with a label $\omega \in \Omega$ if a formula ϕ is valid, locally for the root or globally for any node.
$\mid (a + (\phi, \psi)) \mid (a - (\phi, \psi))$ for adding or removing edges with the feature a, between nodes verifying a formula ϕ and those verifying a formula ψ.
$\mid (\alpha_1; \alpha_2) \mid (\alpha_1 \cup \alpha_2) \mid \alpha_1^*$ for the definition of a sequence, a choice or an iteration over actions α_1 and α_2.

For a formula ϕ:

$$\phi ::= \omega \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid [\alpha]\phi$$

ω as a formula means that node labels can be atomic formulas. And intuitively, $[\alpha]\phi$ means that after any execution of an action α, the formula ϕ holds. The propositional logic of such dynamic logic being a classical one, some more symbols can be defined as usual conjunction, implication and equivalency being respectively $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$, $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$ and $\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$, for given formulas ϕ and ψ. One more modality can be defined, for a given action α and formula ϕ, as $< \alpha > \phi \equiv \neg[\alpha]\neg\phi$. Intuitively, it holds when an execution of the action α makes ϕ hold.

3.2 Semantics of the Dynamic Logic

A semantic can be defined for this dynamic logic using rooted termgraphs as models and verifying the properties expressed by the formulas on them, using the sets of labels and features of the logic to define them. [1]

Models. The rooted termgraphs used here are different from the previous ones. The labeling function for nodes is now defined over the power set of Ω, so typed $\mathcal{L}^{\mathcal{N}} : \mathcal{N} \rightarrow \mathcal{P}(\Omega)$, of what the former definition is said to be a particular case.

Interpretation of Actions and Formulas. Before the definition of a satisfiability relation, it needs interpretation functions over the models.

I_G is defined as

- $I_G(a) = \{e \mid e \in \mathcal{E} \text{ and } \mathcal{L}^{\mathcal{E}}(e) = a\}$ the set of a edges.
- $I_G(\omega) = \{n \mid n \in \mathcal{N} \text{ and } \omega \in \mathcal{L}^{\mathcal{N}}(n)\}$ the set of nodes having the ω label.

And, for any $a \in \mathcal{F}$, R_G is a family of binary relations defined such as

- $R_G(a) = \{(n_1, n_2) \mid \exists e \in I_G(a),\ \mathcal{S}(e) = n_1 \text{ and } \mathcal{T}(e) = n_2\}$

Because of dependency between formulas and actions, the satisfiability relation $\models$ for a formula F and a rooted termgraph G requires an inductive definition on F, dependent of a relation $G \longrightarrow_\alpha G'$ for every action α.

- $G \vDash \omega$ iff $n_0 \in I_G(\omega)$, interpreting ω in G.
- $G \nvDash \bot$
- $G \vDash \neg\phi$ iff $G \nvDash \phi$
- $G \vDash \phi \vee \psi$ iff $G \vDash \phi$ or $G \vDash \psi$
- $G \vDash [\alpha]\phi$ iff for any rooted termgraph G', if $G \longrightarrow_\alpha G'$ then $G' \vDash \phi$

with $G \longrightarrow_\alpha G'$ the binary relation between two termgraphs G and G' considering the action α. It is defined inductively on α, with $G = (\mathcal{N}, \mathcal{E}, \mathcal{L}^{\mathcal{N}}, \mathcal{L}^{\mathcal{E}}, \mathcal{S}, \mathcal{T}, r)$ and $G' = (\mathcal{N}', \mathcal{E}', \mathcal{L}^{\mathcal{N}'}, \mathcal{L}^{\mathcal{E}'}, \mathcal{S}', \mathcal{T}', r')$, but only definitions for the useful cases will be introduced. The definitions are declarative, so it should be read as conditions for a correct relation between G and G' and not as the way to get a model G' from a model G.

A notation $[\![e]\!]_G$ is introduced for a graph G and an edge e of G, to express $(\mathcal{S}(e), \mathcal{L}^{\mathcal{E}}(e), \mathcal{T}(e))$. Ambiguity decreases, in the following definitions, the set $\mathcal{E}$ being only identifiers of edges for the functions $\mathcal{L}^{\mathcal{E}}, \mathcal{S}$ and $\mathcal{T}$ and not tuples of these informations. We can justify this by looking at the other way, with $\mathcal{E}$ as a set of tuples, and seeing that it does not make much sense: for example, when redirecting edges, the set $\mathcal{E}$ was staying the same, and so were the tuples in this set, but the target of an edge was changed and thus a tuple was associated by functions to information which is no longer the content of the tuples.

For convenience, another notation $G_{[n']}$ is introduced for a node n' of a graph $G = (\mathcal{N}, \mathcal{E}, \mathcal{L}^{\mathcal{N}}, \mathcal{L}^{\mathcal{E}}, \mathcal{S}, \mathcal{T}, r)$, to express the graph $(\mathcal{N}, \mathcal{E}, \mathcal{L}^{\mathcal{N}}, \mathcal{L}^{\mathcal{E}}, \mathcal{S}, \mathcal{T}, n')$, or in more simple terms, to express changing the root of G with n'.

$G \longrightarrow_a G'$ iff
- $\mathcal{N}' = \mathcal{N}$, $\mathcal{E}' = \mathcal{E}$, $\mathcal{L}^{\mathcal{N}'} = \mathcal{L}^{\mathcal{N}}$, $\mathcal{L}^{\mathcal{E}'} = \mathcal{L}^{\mathcal{E}}$, $\mathcal{S}' = \mathcal{S}$ and $\mathcal{T}' = \mathcal{T}$, to express almost all parts of G' being the same.
- $(n_0, n_0') \in R_G(a)$, to express the possibility to navigate from the root of G to the root of G'.

$G \longrightarrow_U G'$ iff
- $\mathcal{N}' = \mathcal{N}$, $\mathcal{E}' = \mathcal{E}$, $\mathcal{L}^{\mathcal{N}'} = \mathcal{L}^{\mathcal{N}}$, $\mathcal{L}^{\mathcal{E}'} = \mathcal{L}^{\mathcal{E}}$, $\mathcal{S}' = \mathcal{S}$ and $\mathcal{T}' = \mathcal{T}$, to express no characterization for the root of G' but other parts being the same.

$G \longrightarrow_{(\omega :=_g \phi)} G'$ iff
- $\mathcal{N}' = \mathcal{N}$, $\mathcal{E}' = \mathcal{E}$, $\mathcal{L}^{\mathcal{E}'} = \mathcal{L}^{\mathcal{E}}$, $\mathcal{S}' = \mathcal{S}$, $\mathcal{T}' = \mathcal{T}$ and $r' = r$
- for any $m \in \mathcal{N}$, if $G_{[m]} \vDash \phi$ then $\mathcal{L}^{\mathcal{N}'}(m) = \mathcal{L}^{\mathcal{N}}(m) \cup \{\omega\}$ else $\mathcal{L}^{\mathcal{N}'}(m) = \mathcal{L}^{\mathcal{N}}(m) \backslash \{\omega\}$, expressing the addition or deletion of the label ω of any node m of the graph satisfying the formula ϕ.

$G \longrightarrow_{\phi?} G'$ iff

- $\mathcal{N}' = \mathcal{N}$, $\mathcal{E}' = \mathcal{E}$, $\mathcal{L}^{\mathcal{N}'} = \mathcal{L}^{\mathcal{N}}$, $\mathcal{L}^{\mathcal{E}'} = \mathcal{L}^{\mathcal{E}}$, $\mathcal{S}' = \mathcal{S}$, $\mathcal{T}' = \mathcal{T}$ and $r' = r$
- $G \vDash \phi$, to express the formula ϕ being valid for G.

$G \longrightarrow_{(\omega:=l\,\phi)} G'$ iff
- $\mathcal{N}' = \mathcal{N}$, $\mathcal{E}' = \mathcal{E}$, $\mathcal{L}^{\mathcal{E}'} = \mathcal{L}^{\mathcal{E}}$, $\mathcal{S}' = \mathcal{S}$, $\mathcal{T}' = \mathcal{T}$ and $r' = r$
- if $G \vDash \phi$ then $\mathcal{L}^{\mathcal{N}'}(r) = \mathcal{L}^{\mathcal{N}}(r) \cup \{\omega\}$ else $\mathcal{L}^{\mathcal{N}'}(r) = \mathcal{L}^{\mathcal{N}}(r) \backslash \{\omega\}$, to express the addition or deletion of the label ω from the root.

$G \longrightarrow_{(a+(\phi,\psi))} G'$ iff
- $\mathcal{N}' = \mathcal{N}$ and $\mathcal{L}^{\mathcal{N}'} = \mathcal{L}^{\mathcal{N}}$
  Considering the set of candidate edges $\mathcal{C} = \{(n_s, a, n_t) \mid$ with $n_s \in \mathcal{N}$ and $n_t \in \mathcal{N}$ such as $G_{[n_s]} \vDash \phi$ and $G_{[n_t]} \vDash \psi\}$, to be added only between nodes validating the formulas ϕ and ψ.
- $\mathcal{E}' \supset \mathcal{E}$, and for all $e \in \mathcal{E}, [\![e]\!]_{G'} = [\![e]\!]_G$, characterizing an addition without any loss.
- for all $p \in \mathcal{C}, \exists e \in \mathcal{E}', [\![e]\!]_{G'} = p$ and for all $e \in \mathcal{E}' \setminus \mathcal{E}$, $[\![e]\!]_{G'} \in \mathcal{C}$, expressing candidate edges being added in $\mathcal{E}'$ but nothing else.

$G \longrightarrow_{(a-(\phi,\psi))} G'$ iff
- $\mathcal{N}' = \mathcal{N}$ and $\mathcal{L}^{\mathcal{N}'} = \mathcal{L}^{\mathcal{N}}$
  Considering the set of deleted edges $E = \{e \mid e \in \mathcal{E}$ such as $[\![e]\!]_{G'} = (n_s, a, n_t)$ with n_s and n_t such as $G_{[n_s]} \vDash \phi$ and $G_{[n_t]} \vDash \psi\}$.
- $\mathcal{E}' = \mathcal{E} \setminus E$ characterizing deletion of edges only between nodes validating formulas ϕ and ψ.
- for all $e \in \mathcal{E}$, $\mathcal{L}^{\mathcal{E}'}(e) = \mathcal{L}^{\mathcal{E}}(e)$, $\mathcal{S}'(e) = \mathcal{S}(e)$, $\mathcal{T}'(e) = \mathcal{T}(e)$ and $r = r'$.

$G \longrightarrow_{\alpha;\beta} G'$ iff
- there exists a rooted termgraph G'' such $G \longrightarrow_{\alpha} G''$ and $G'' \longrightarrow_{\beta} G'$.

$G \longrightarrow_{\alpha*} G'$ iff
- there is a rooted termgraph sequence $(G^{(0)},\ldots,G^{(k)})$ with $G^{(0)} = G$, $G^{(k)} = G'$ and for all $i \in \{0, \ldots, k-1\}$, $G^{(i)} \longrightarrow_{\alpha} G^{(i+1)}$.

Semantics of left over actions $n, \boldsymbol{n}$ and $\alpha_1 \cup \alpha_2$ are in the original paper.
At this point, the logic allows to characterize classes of graphs using the satisfiability of formulas by these graphs as models. Everything goes pretty well, but issues come when dealing with rewriting and propositions made to talk about graph rewriting.

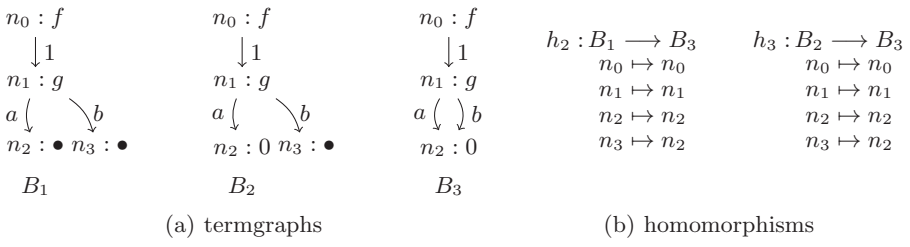
4 Actual Rewriting, and its Issues

The approach here is an algorithmic one: transformation actions are defined within a rewriting system and can be applied to a graph, alone or sequentially. It forms rules of rewriting that could be applied if an instantiation of the rule is found in a given graph. Such instance can be defined with a graph morphism which embeds the graph domain of the rule into the graph in which the rule could be applied.

4.1 Homomorphism of Graphs

A homomorphism of labelled graphs $h : G \to G'$ can be defined, given two rooted termgraphs $G = (\mathcal{N}, \mathcal{E}, \mathcal{L}^{\mathcal{N}}, \mathcal{L}^{\mathcal{E}}, \mathcal{S}, \mathcal{T}, r)$ and $G' = (\mathcal{N}', \mathcal{E}', \mathcal{L}^{\mathcal{N}'}, \mathcal{L}^{\mathcal{E}'}, \mathcal{S}', \mathcal{T}', r')$. Somewhat, only with a function $h^n : \mathcal{N} \to \mathcal{N}'$ preserving the labeling of nodes but equally preserving the source and target function for the edges, and thus the labelization of edges.
So $\forall m \in \mathcal{N}, \mathcal{L}^{\mathcal{N}'}(h^n(m)) = \mathcal{L}^{\mathcal{N}}(m)$ is mandatory and then because of the determinism condition satisfied by the graphs, there is no ambiguity on the conditions for the edges of the codomain graph G'. For any e of $\mathcal{E}$, it only requires one existing e' of $\mathcal{E}'$ verifying $\mathcal{S}'(e') = h^n(\mathcal{S}(e))$, $\mathcal{L}^{\mathcal{E}'}(e') = \mathcal{L}^{\mathcal{E}}(e)$ and $\mathcal{T}'(e') = h^n(\mathcal{T}(e))$, and so with corresponding source and target while preserving the edge labelization, mandatory too for such homomorphism. There is no specific condition for correspondence of roots of the two termgraphs.

Examples of the original paper can be presented here, in Figure 2, to display some homomorphisms: morphisms h_2 and h_3, between three graphs $B1$, $B2$ and $B3$ displaying the association of their nodes.



(a) termgraphs (b) homomorphisms

Fig. 2. Examples of termgraph homomorphisms

Existence of Graph Homomorphisms and Particularity of such Homomorphisms

In the original paper, a way to talk about homomorphisms on graphs using the logic is proposed, a formula express this concept and is defined for a graph $G = (\mathcal{N}, \mathcal{E}, \mathcal{L}^{\mathcal{N}}, \mathcal{L}^{\mathcal{E}}, \mathcal{S}, \mathcal{T}, r)$. Thus an action α_G and a formula ϕ_G can relate that there is a homomorphism from G to a graph G' if and only if $G' \vDash < \alpha_G > \phi_G$.[1]

For this, considering $\mathcal{N} = \{n_0, \ldots, n_{N-1}\}$, with N being the number of nodes of G and n_0 being its root, and considering a sequence $P = \{\pi_0, \ldots, \pi_{N-1}\}$ of distinct elements of Ω, each π_i is going identify the node n_i.

The action α_G is defined

- with $\beta_G = (\pi_0 :=_g \bot) ; \ldots ; (\pi_i :=_g \bot) ; \ldots ; , (\pi_{N-1} :=_g \bot)$, characterizing the elimination of any label π_i.

- with, for $0 \leq i \leq N-1$, $\gamma_G^i = (\neg\pi_0 \wedge \ldots \wedge \neg\pi_i)?\,;\,(\pi_i :=_l \top)\,;\,U$, characterizing the labelization with π_i of a node not already labelled with π_k for $k \leq i$.
- and finally $\alpha_G = \beta_G\ ;\ \gamma_G^0\ ;\ldots;\ \gamma_G^{N-1}$, sequencing these actions.

Then, the formula ϕ_G is defined

- with $0 \leq i \leq N-1$, if $\mathcal{L}^{\mathcal{N}}(n_i) \neq \emptyset$ then $\psi_G^i =< U > (\pi_i \wedge \mathcal{L}^{\mathcal{N}}(n_i))$ else $\psi_G^i = \top$, characterizing the conservation of the labelization of a node identified as the image of n_i, by the label π_i, but nothing if this node wasn't labelled.
- with for $0 \leq i, j \leq N-1$, if $\exists e \in \mathcal{E}$ such $\mathcal{S}(e) = n_i$ and $\mathcal{T}(e) = n_j$ then $\zeta_G^{i,j} =< U > (\pi_i \wedge < \mathcal{L}^{\mathcal{E}}(e) > \pi_j)$ else $\zeta_G^{i,j} = \top$, characterizing the existence of edges corresponding to the ones of G.
- and finally $\phi_G = \psi_G^0 \wedge \ldots \wedge \psi_G^{N-1} \wedge \zeta_G^{0,0} \wedge \ldots \wedge \zeta_G^{N-1,N-1}$, verifying all these formulas.

In the latter definition, it is assumed to have a subset P of Ω of fresh labels not already used in G, and so that do not have to be preserved by the homomorphism in G', and do not erase information when used in β_G. Such dedicated labels for identification of the elements of $\mathcal{N}$ could be assumed as a part of the set Ω, by definition. This definition of P, identifying differently each node of G and the way it is used in the formula, requires more for a homomorphism than what implies the homomorphism definition. Acutally, the examples of the Figure 2, which do not stand against the definition, are not injective morphisms. In the formula $< \alpha_G > \phi_G$, matched nodes are labelled to be identified to only one node of G, as the test $(\neg\pi_0 \wedge \ldots \wedge \neg\pi_i)?$ express it. Therefore, it is mandatory for a homomorphism $h : G \longrightarrow G'$ to be injective to satisfy the formula and this necessary condition on the models. Although, such injectivity is a common feature of graph morphism in many graph rewriting frameworks, so it is not mandatory to define formally such a system using injective matching only and the initial definition of graph homomorphism should specify wether this particularity is actually required.

4.2 Rewriting Step and Translation in Logic

The rewriting is defined to be applied to a graph $G = (\mathcal{N}, \mathcal{E}, \mathcal{L}^{\mathcal{N}}, \mathcal{L}^{\mathcal{E}}, \mathcal{S}, \mathcal{T}, r)$ to obtain a graph $G' = (\mathcal{N}', \mathcal{E}', \mathcal{L}^{\mathcal{N}'}, \mathcal{L}^{\mathcal{E}'}, \mathcal{S}', \mathcal{T}', r')$, the result of the application of an action α to a graph G is denoted $\alpha[G]$. It is possible to make sequences of actions, as empty or the concatenation of an action α and another sequence, and the application of a sequence Δ to a graph G is denoted $\Delta[G]$.[1]

- if Δ is the empty sequence then $\Delta[G] = G$
- if $\Delta = \alpha; \Delta'$ is a concatenation, with a sequential operator ";", then $\Delta[G] = \Delta'[\,\alpha[G]\,]$

For a morphism h and a sequence Δ, $h(\Delta)$ denotes the sequence one obtains by substitution in Δ of any node n by $h(n)$.

The original paper proposes a way to talk about sequences of actions, by describing the sequence of rewriting actions as a sequence of logical actions. Thus, after the translation of an action α of the rewriting system, the relation $\longrightarrow_{tr(\alpha)}$ introduced with the semantic of the logic allows to relate models, the second potentially being the result of the application of the action to the first one. The translation of a sequence of a given action a and another sequence Δ will be the sequence of translations $\alpha_{a;\Delta} = \alpha_a; \alpha_\Delta$. Assuming that any action has a translation entirely independent of the rest of the sequence, the translation order actually does not matter, but a sequential translation seizes a correct idea.

For the rewriting actions:

$n \gg_a m$ is a local redirection.
An outgoing edge from a node n with the feature a is modified to point to a node m.

- $\mathcal{N}' = \mathcal{N}, \mathcal{E}' = \mathcal{E}, \mathcal{L}^{\mathcal{N}'} = \mathcal{L}^{\mathcal{N}}, \mathcal{L}^{\mathcal{E}'} = \mathcal{L}^{\mathcal{E}}, \mathcal{S}' = \mathcal{S}$ and $r = r'$, the nodes, edges, their labelization, the source of the edges and the root are the same.
- for $e \in \mathcal{E}$ such $\mathcal{S}'(e) = n$ and $\mathcal{L}^{\mathcal{E}'}(e) = a$ then $\mathcal{T}'(e) = m$, the target of the only wanted edge is changed for m. $\forall e' \in \mathcal{E}'$, if $e' \neq e$ then $\mathcal{T}'(e') = \mathcal{T}(e')$, the target function doesn't change for the other edges.

To begin with, here is the given formula for the local redirection.

- $\alpha_{n \gg_a m} = (a - (\pi_n, \top))\,;\,(a + (\pi_n, \pi_m))$

One can notice that this formula depicts a transformation by deleting an edge, labelled a, between a node n and any other node, though the determinism dictates the existence of at most only one such edge. But above all thus an a edge between this node n and a node m is added at the end. Looking at the definition of the local redirection, one can see that $\mathcal{E}' = \mathcal{E}$ specify that no edge is actually added and the other item clearly specify that only an existing e of E with source as n and feature as a has its target changed to m.

A difference happens between the rewriting of a graph and the models linked by the actions of the translation of the rewriting action, as shown in Figure 3 which is a counter example for adequation between this translation and rewriting.

(a) rewrite rule (b) models

Fig. 3. Local Redirection Difference

The problem being that the logical actions don't require the existence of the redirected edge, what is obviously mandatory to rewrite it. We proposed, as a correction for this problem, the formula:

– $\alpha_{n \gg_a m} = (\lambda_n :=_g \bot) ; (\lambda_n :=_g \pi_n \wedge < a > \top) ; (a-(\pi_n, \top)) ; (a+(\lambda_n, \pi_m))$

The added actions require the marking of a node n with a λ_n, assumed fresh for the graph, validating the possibility to navigate along a a edge to any node, then it proceeds to delete and add the edge only with this sole mark. If no edge was removed, no mark will be found on the node n, correcting this problem of adequation between rewriting and logical translation.

$n \gg m$ is a global redirection.
The target of every edge pointing to the node n is modified to make the edges point to the node m.

– $\mathcal{N}' = \mathcal{N}$, $\mathcal{L}^{\mathcal{E}'} = \mathcal{E}$, $\mathcal{L}^{\mathcal{N}'^{\prime}} = \mathcal{L}^{\mathcal{N}}$, $\mathcal{L}^{\mathcal{E}\prime} = \mathcal{L}^{\mathcal{E}}$ and $\mathcal{S}' = \mathcal{S}$, the nodes, edges, their labelizations and the source of the edges.
– for all $e \in \mathcal{E}$ such as $\mathcal{T}(e) = n$ then $\mathcal{T}'(e) = m$ else $\mathcal{T}'(e) = \mathcal{T}(e)$, the targets of the only wanted edges are changed for m, otherwise it does not change.
– if $n = r$ then $r' = m$ else $r' = r$, the root changes if it's the former target of the changed edges.

If one looks at the translation of the global redirection, with first a formula to globally redirect to a node m every a edges initially pointing to a node n:

– $\alpha_{n \gg^g_a m} = (\lambda_a :=_g \bot) ; (\lambda_a :=_g < a > \pi_n) ; (a - (\top, \pi_n)) ; (a + (\lambda_a, \pi_m))$

This formula does not suffer of this problem, the second action marking with a λ_a any reachable node by an edge with the feature a. Then, the action adding the edges does not add wrongly an inexistant previously removed edge, since the label λ_a ensures these nodes are subjects to redirection. However, it is implicitly required that this λ_a is a dedicated label for this action, to not interfere with any other formula using this label for another purpose. The full translation of the rewriting step is finally a sequence of the previously defined formulas, for every feature of the graph:

– $\alpha_{n \gg m} = ;_{a \in \mathcal{F}} \alpha_{n \gg^g_a m}$

Comes finally the last rewriting action, the node labelization.

$n : \omega\,(f_1 \Rightarrow n_1, \ldots, f_k \Rightarrow n_k)$ is a node definition or labelization.
It adds a node n, if it does not already belongs to the graph, or modifies an already existing one. It assigns the label ω and defines the edges $e_1\, to\, e_k$ outing of this node n, respectively pointing to the nodes $n_1\, to\, n_k$ with the labels $f_1\, to\, f_k$, according to the following definition:

– $\mathcal{N}' = \mathcal{N} \cup \{n, n_1, ..., n_k\}$, nodes of the rules which are not already included in G are added.

- $\mathcal{L}^{\mathcal{N}''}(n) = \omega$ and $\forall m \in \mathcal{N} \setminus \{n\}$, $\mathcal{L}^{\mathcal{N}'}(m) = \mathcal{L}^{\mathcal{N}}(m)$, n is labelled with ω and the other nodes keep the same labeling.
  Considering the newly defined edges $E = \{e_i | \mathcal{S}'(e_i) = n,\ \mathcal{L}^{\mathcal{E}'}(e_i) = f_i$ and $\mathcal{T}'(e_i) = n_i\}$ with $1 \leq i \leq k$.
- $\mathcal{E}' = \mathcal{E} \cup E$, new edges are added to the already existing ones.
- $\forall e_i \in E, \mathcal{L}^{\mathcal{E}'}(e_i) = f_i$, the features of the new edges are defined, and $\forall e \notin E, \mathcal{L}^{\mathcal{E}'}(e) = \mathcal{L}^{\mathcal{E}}(e)$, the features of the other edges don't change.
- $\forall e_i \in E, \mathcal{S}'(e_i) = n$ and $\forall e \notin E, \mathcal{S}'(e) = \mathcal{S}(e)$, the same thing is done for the source function.
- $\forall e_i \in E, \mathcal{T}'(e_i) = n_i$ and $\forall e \notin E, \mathcal{T}'(e) = \mathcal{T}(e)$, the same thing is done for the target function.
- $r' = r$, the root remains the same.

And the formula, initially given as its translation:

- $\alpha_{n:\omega\,(f_1 \Rightarrow n_1, \ldots, f_k \Rightarrow n_k)} =$

$$U\,;\,\pi_n?\,;\,(\omega :=_l \top)\,;\,(f_1 + (\pi_{n,}, \pi_{n_1}))\,;\,\ldots\,;\,(f_k + (\pi_{n,}, \pi_{n_k}))$$

One could there raise a first issue, regarding the end of the formula: it adds without much care outgoing edges of the node n, thereby not ensuring the determinism. The result is that no model can be related as resulting of the application of this action, when what was intended was a relation on models displaying a redirection of these edges. We proposed this correction, using as previously the idea of edge redirection by deletion of the former edge and addition of the redirected one.

- $\alpha_{n:\omega\,(f_1 \Rightarrow n_1, \ldots, f_k \Rightarrow n_k)} =$

$$U\,;\,\pi_n?\,;\,(\omega :=_l \top)\,;$$
$$(f_1 - (\pi_n, \top))\,;\,\ldots\,;\,(f_k - (\pi_n, \top))\,;\,(f_1 + (\pi_n, \pi_{n_1}))\,;\,\ldots\,;\,(f_k + (\pi_n, \pi_{n_k}))$$

Now, a model related by the relation of the corresponding formula for this action can exist, and it will be a deterministic graph with redirected edges. However, there still remains a problem. If one looks at the transformation expressed by the formula, it translates the labelization of a node positionning the root on a node n, identified by π_n, thus it labelizes it with ω and finally changes the edges outgoing of this node n. As displayed in Figure 4, this raises again a major difference between the graph one obtains by rewriting and the models related by the relation for the modality of the logical action. The behavior is not the same because the formula requires only the addition of a label to the multiple already existing labels of a node while the rewriting action requires the replacement as an unique label.

For a better understanding of the extent of this problem, one should look at the following, which demonstrates how much of an issue that becomes in regard of characterization of rewriting system.

(a) rewrite rule (b) models

Fig. 4. Node Labeling Difference

4.3 Rewriting Rules and Rewriting System Characterization

Rewriting rules

A rewriting rule is expressed as a graph L and a sequence of actions Δ to apply, and will be noted (L, Δ). It is said that a graph G is rewritten as a graph G' if there exists a homomorphism $h : L \to G$ such as $h(\Delta)[G] = G'$, denoted $G \to_{L,\Delta} G'$.
A simple yet useful example may be defined, representing an action on a simple on-off switch of an electrical network:

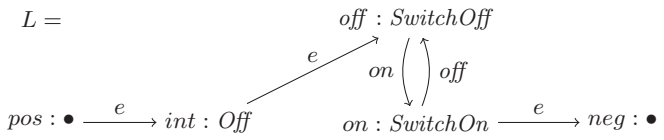


Fig. 5. Left-hand side L of a rule

Displayed in Figure 5, on the left of L there is a positive terminal of the circuit, on the right side there is a negative one. The edges labelled edges as e are for the electrical circuit and the switch is currently set to *Off*. The two positions *SwitchOn* and *SwitchOff* are linked to each other to avoid any mismatch if another switch is part of the network, even though a one-way link alone could be enough.

$$\Delta_{switchOn} = (int : On\ (\ )\ ;\ int >>_e on)$$

Fig. 6. Sequence Δ of a rule

The sequence for this rule, as displayed in Figure 6, is made of an action to mark the switch node with *On* without adding any extra edge, and an action to redirect the edge e, outgoing from this node to the Off position of the switch, to get a graph representing a closed electrical circuit, displayed now in Figure 7 as the result on the graph L.

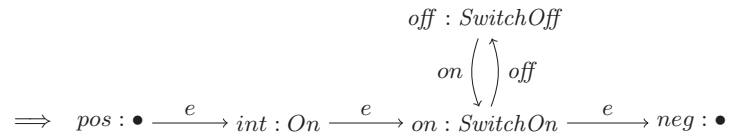


Fig. 7. result of applying Δ to L

Rewriting System Characterization and The Labelization Problem

Using the previous example of a switch activation, a graph G is displayed in the Figure 8, being a simple electrical network having only a generator and a switch, and being really similar to the left-hand side of the rule $(L, \Delta_{switchOn})$ previously defined.

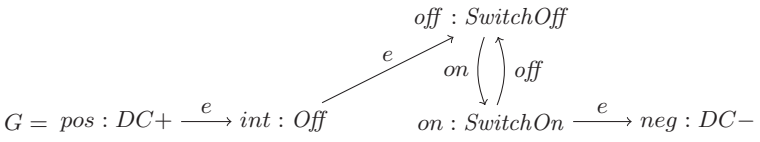


Fig. 8. graph G

The normal form of the graph G, with respect to the rule $(L, \Delta_{switchOn})$, satisfies the formula $\psi := [U]\neg Off$, because of its switch being activated and thus being labelled with On, so no node is labelled with Off anymore.

In the end of the original paper, the proposition 4 [1] which defines a way to talk about a normal formal of a graph G satisfying ψ says that this should be equivalent to the following satisfaction

$$G \vDash [(\alpha_L; \phi_L?; \alpha_{\Delta_{switchOn}})^*]([\alpha_L; \phi_L?] \perp \rightarrow \psi)$$

To explain this proposition, one can think about a normal form with respect of a rule as resulting of successive matchings and applications of this rule. When the rule cannot be matched anymore this implies that the formula ψ holds, which characterizes the normal forms. In the example with the rule $(L, \Delta_{switchOn})$, every switch must be activated and the rule shouldn't be matchable then, because there is no label Off or any e edge pointing to the right position. But because of the difference between rewriting and translation in the logic, when every e edge of the switch is redirected in the models, no more matching is available but there is still the label Off, along with the label On labeling the node int. The formula ψ is thus unsatisfied, the graph G and the formula ψ are a counter example for this proposition.

First, one can notice that relying on the definition of the models, the logic makes use of the labels to mark and reason, but without any difference between the ones being actual parts of the graph and the ones serving the logic. Even if the problem appears with the semantics, the solution is not there, because any change of the model definition, allowing to make a difference between graph and logic for example by simply splitting in two layer these informations, doesn't mean that the same syntax of the logic allows to express this difference.

The problem comes with the translation of the label definition of a node, it seems sounding to complete this translation to get the correspondence between rewriting and relation on models. But the rules and the matching use the labels to logically identify the node, so one should be careful with erasing, thus to erase everything is not an option here. Actually, the syntax of the logic needs explicitly the label to delete, with a global or local labeling action. And the syntax of the action currently does not allow to say which label will be erased. This is where the problem lies in: there is only the identifier of a node and the new label.
To actually translate the action, it is possible if it is given more information, so a new label definition action is a possibility or the syntax of the logic should be changed to express actions on a new model type.

But before any syntactic change, of rewriting actions or logic, it is interesting to consider the action in the context of a graph. Because of the $\mathcal{L}^{\mathcal{N}} : \mathcal{N} \longrightarrow \Omega$ of the graph, and having already the node n of the action, the labeling function can give the label to remove. Regarding of the action to translate, there is not this information, but more generally rewriting actions are defined for a specific graph. Thus, lifting the translation in formula to the level of the rule, this gives the required information. From this rule, a graph is given as left-hand side and thus a $\mathcal{L}^{\mathcal{N}} : \mathcal{N} \longrightarrow \Omega$ function can be used, and it allows an explicit erasing of a label as it was implicitly expected by the rewriting action.

For a graph $G = (\mathcal{N}, \mathcal{E}, \mathcal{L}^{\mathcal{N}}, \mathcal{L}^{\mathcal{E}}, \mathcal{S}, \mathcal{T}, r)$ and a node labeling action $n : \omega\,(f_1 \Rightarrow n_1, \ldots, f_k \Rightarrow n_k)$, we give as a correct translation the formula

$$- \ \alpha_{G,\, n : \omega\,(f_1 \Rightarrow n_1, \ldots, f_k \Rightarrow n_k)} =$$

$$U \ ; \ \pi_n? \ ; \ (\mathcal{L}^{\mathcal{N}}(n) \ :=_l \bot) \ ; \ (\omega \ :=_l \top) \ ;$$
$$(f_1 - (\pi_n, \top)) \ ; \ \ldots \ ; \ (f_k - (\pi_n, \top)) \ ; \ (f_1 + (\pi_n, \pi_{n_1})) \ ; \ \ldots \ ; \ (f_k + (\pi_n, \pi_{n_k}))$$

However, the application of the rule to the left-hand side results in another graph with another labeling function and the translation of a sequence cannot remain as previously proposed. Every node labeling action should be translated depending on the result of previously applied actions. The translation of a sequence of an action a and another sequence Δ is a sequence of sequential translations:

$$\alpha_{G,\,(a;\Delta)} = \alpha_{G,\,a}; \alpha_{a[G],\,\Delta}$$

One could notice that this translation is still done in a linear way through the sequence, but this definition could actually be slightly changed because every action but node labelization can be translated independently. Only the node labelization is dependent of a context and more precisely there are conflicts only

when editing the same node, having to erase a label which was just defined previously in the sequence.

5 Conclusion

This paper provided an introduction to a dynamic logic, defined by P. Balbiani, R. Echahed and A. Herzig. Originally, a formalization was done using the Coq proof assistant and so was done a study of this logic. During this work some mistakes were spotted. While sometimes these were just minor-looking imprecisions, when working with formal logic, it may be relevant to point out such ambiguities if the logic can become incoherent with a wrong interpretation.

Other mistakes are not only due to interpretation, and as it was demonstrated, they allow to find counter-examples for propositions and thus to establish incoherence with the main goal of this logic, to talk about rewriting systems. Solutions to these issues are proposed, they resolve the issues by getting back to an adequation between actual rewriting and relations on models of the logic.

A first idea to find another technical problem is the conservation of determinism of graphs, which only seems currently assumed and can be broken by the action of the logic, heading to the impossibility to relate a model to another because of the determinism, assumed by definition. It could be more explicit in the semantics of rewriting, and currently the definition of one of the rewriting actions allows to add edges breaking this condition. This is something the logic can express and handle during the translation of these rewriting actions, this looks like an interesting improvement of the logical framework.

The models are defined with a multi-labeling function of nodes and already demonstrate some differences during translation of rewriting into the logic, because the models are graphs with a unique-labeling function into the initial rewriting system. It is not totally clear whether the difference stops there, and it is required to study more uses of the logic in regards of rewriting to be sure of the correct use. It seems interesting to study the reverse translation as well, from logic to rewriting, to get an useful and complete logical framework, in the idea of $Curry - Howard$ correspondence with terms as proofs.

References

1. Balbiani, P., Echahed, R., Herzig, A.: A dynamic logic for termgraph rewriting. In Ehrig, H., Rensink, A., Rozenberg, G., Schürr, A. (eds.) ICGT. LNCS, vol. 6372, pp.59–74. Springer, Heidelberg (2010)
2. Ariola, Z.M., Klop, J.W.: Lambda calculus with explicit recursion. Inf. Comput. 147(2), 154–233 (1997)
3. Barendregt, H., van Eekelen, M., Glauert, J., Kennaway, J., Plasmeijer, M., Sleep, M.: Term graph rewriting. In de Bakker, J., Nijman, A., Treleaven, P. (eds.) PARLE Parallel Architectures and Languages Europe. LNCS, vol. 259, pp. 141–158. Springer, Heidelberg (1987)
4. Harel, D., Kozen, D., Tiuryn, J.: Dynamic logic. Handbook of Philosophical Logic, MIT Press. 497–604 (1984)